NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

MULTI-FRAME OBJECT DETECTION

by

Michael J. Laielli

September 2012

Thesis Advisor:	Mathias Kolsch
Second Reader:	Richard C. Olsen

Approved for public release; distribution is unlimited

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704–0188

1. REPORT DATE *(DD–MM–YYYY)* 24–9–2012	**2. REPORT TYPE** Master's Thesis	**3. DATES COVERED** *(From — To)* 2011-09-01—2012-9-28

4. TITLE AND SUBTITLE

Multi-frame Object Detection

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Michael J. Laielli

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Naval Postgraduate School
Monterey, CA 93943

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Department of the Navy

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited

13. SUPPLEMENTARY NOTES

The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.IRB Protocol Number: N/A

14. ABSTRACT

This thesis describes an object detection system that extracts and combines appearance information over multiple consecutive video frames, inherently gaining and analyzing information related to motion. Objects that exhibit characteristic motion over the course of multiple frames are able to be detected at smaller scales than achievable by a single-frame detector. Our method builds on the detection work of Viola and Jones, with our extension being the added ability to combine information from multiple frames. Our implementation detects an object in synthetic images at very small scales, down to 3×3 pixels, and has a low false-alarm rate.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**			**19b. TELEPHONE NUMBER** *(include area code)*
Unclassified	Unclassified	Unclassified	UU	45	

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

MULTI-FRAME OBJECT DETECTION

Michael J. Laielli, Research Assistant, Remote Sensing Center, Naval Postgraduate School

B.S., Computational Science, The Richard Stockton College of New Jersey, 2010

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN REMOTE SENSING INTELLIGENCE

from the

NAVAL POSTGRADUATE SCHOOL
September 2012

Author:	Michael J. Laielli
Approved by:	Mathias Kolsch Thesis Advisor
	Richard C. Olsen Second Reader
	Dan Boger Chair, Department of Information Science

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

This thesis describes an object detection system that extracts and combines appearance information over multiple consecutive video frames, inherently gaining and analyzing information related to motion. Objects that exhibit characteristic motion over the course of multiple frames are able to be detected at smaller scales than achievable by a single-frame detector. Our method builds on the detection work of Viola and Jones, with our extension being the added ability to combine information from multiple frames. Our implementation detects an object in synthetic images at very small scales, down to 3×3 pixels, and has a low false-alarm rate.

THIS PAGE INTENTIONALLY LEFT BLANK

Table of Contents

1 Introduction **1**

1.1 Objective 2

1.2 Terminology 2

2 Background **5**

2.1 Viola-Jones Visual Object Detection Framework 5

2.2 Software Used 9

3 Method **11**

3.1 Implementing Multi-frame Feature Calculations 12

4 Experiment **17**

4.1 Dataset 17

4.2 Training the Detectors 17

5 Results **21**

5.1 ROC Curve 21

5.2 Weak Classifiers per Stage 22

6 Discussion **25**

6.1 Training Results 25

6.2 Limitations and Future Work 26

7 Conclusion **27**

List of References 29

Initial Distribution List 31

List of Figures

Figure 2.1 An illustrated example of Haar features. 6

Figure 2.2 An illustrated example of integral images. 7

Figure 2.3 An illustrated example of an integral image. Area D can be computed in only four matrix references, $4-3-2+1$, which effectively subtracts areas A, B, and C from the area that extends from the top-left corner to point 4, leaving area D. Points 2 and 3 contain the values of $B+A$ and $C+A$, respectively. Thus, area A is effectively subtracted twice, hence, needs to be added back, once. 8

Figure 2.4 An illustrated example of a cascade of strong classifiers. 10

Figure 3.1 An illustrated example of a feature applied to a multi-frame sample . . 12

Figure 4.1 Examples of single- and multi-frame samples. The top row, left of the line, contains the original-size, 30×30, multi-frame sample, with the scaled-down, 3×3, version in the next row, down. The last two rows show the corresponding single-frame sample at 30×30 and 3×3 pixels. All frames right of the line are displayed to demonstrated the variability between samples. 18

Figure 5.1 ROC curve for single- and multi-frame detector performance on a training set with a sample frame-size of 3×3 pixels. 22

Figure 5.2 FA rate vs weak classifiers (WCs) per stage required in the cascade with a sample frame-size of 3×3 pixels. A perfect final hit rate of 1.0 was held constant. 23

THIS PAGE INTENTIONALLY LEFT BLANK

Acknowledgements

To my wife, Ashlee, I extend a loving thank you for your infinite love, patience, and support. These things from you will always the the foundations of my accomplishments.

To my thesis advisor and Computer Vision Professor, Mathias Kolsch, I extend an inspired thank you for the opportunity to work on an interesting, challenging, and ultimately rewarding project. I've learned more than I could have imagined and I will continue to be inspired by your drive, leadership, and ingenuity.

To my second advisor and Remote Sensing Professor, Richard C. Olsen, I extend a sincere thank you for conceiving and building the Remote Sensing Intelligence program. This rare opportunity has brought me into an exciting new career, for which I will always be grateful.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 1: Introduction

Automatic detection of target objects in full motion video is highly valuable, mainly because manual detection, although more effective, is very inefficient in-terms of human resources. Employing analysts to sort through hours of video footage, or continuously watch a live feed, searching for a specific target is costly and wasteful, especially since the object they are looking for is sparse in many cases. For instance, depending on the dataset, video samples captaining faces, AK-47's, or pedestrians can occur far less frequently than video samples *not* containing such objects. Effectively automating the process saves time and resources.

The beauty of object detection is that it is astoundingly simple to create a detector that never misses a target object. Have it classify everything as positive! While that detector may not be very useful, one that never misses a target and correctly identifies, say, half of the negative data *is* useful. Half of eight hours of negative footage is four hours of negative footage that a human analyst does not have to analyze. Fortunately, current object detectors do much better than a 50 percent false-alarm rate, while maintaining near-perfect detection rates. However, this has not yet been proven at extremely small scales, smaller than 5×5 pixels.

The seminal work of Viola and Jones [1] achieved robust and accurate face detection at 24×24 pixels, while maintaining high speeds. The problem with object detection at extremely small scales is that there simply may not be enough information on a target object to distinguish it from the background. Their later work [2], which accessed information from two frames instead of one, drastically improved the ability to detect pedestrians, but only brought the minimum target resolution down to 20×15 pixels. Others have utilized silhouette tracking over many frames to analyze patterns of motion, improving human action recognition, but not focused on reducing the minimum target resolution [3], [4]. There may be enough information contained within two to five frames to lower the minimum scale of object detection, yet not incur significant degradation of performance. The additional calculations required for multiple frames are at least partly countered by the smaller scale.

This thesis describes a detector that gathers intensity and motion information among multiple frames to make detections. Similarly to the Viola-Jones pedestrian detector [2], short sequences of motion combined with appearance information are extracted to make detections. Our detector extracts this information over more frames, and analyzes the raw pixel intensities, instead of pre-computing difference images.

1.1 Objective

The primary objective of this thesis is to extend a Viola-Jones detection framework implementation to extract and combine intensity information over multiple frames and demonstrate that it outperforms the original single-frame version at extremely small scales.

1.1.1 Hypothesis

Gathering information from multiple consecutive frames enables effective object detection at scales smaller than can be achieved by a single-frame detector.

1.2 Terminology

1.2.1 Image Forms

Throughout this thesis, the terms "sample," "frame," "image," "window," and "background" are designated to refer to certain forms of images in the contexts of the Viola-Jones object detection framework and OpenCV haartraining module. Their definitions are listed here. It is recommended that they are referred back to by the reader when necessary, as their distinctions are important.

Frame A frame *is* a sample, in the case of a single-frame detector. Multiple frames may combine to make an individual sample, as in the case of a multi-frame detector.

Sample The unit of information that is classified. It can be either a single-frame or multi-frame sample.

Single-frame sample A sample containing only one frame.

Multi-frame sample A sample that is a combination multiple consecutive frames.

Image A full-size image, usually cropped from a video file. Samples are extracted from images, or sequences of images.

Image sequence A sequence of images, usually cropped from a video file. Multi-frame samples are extracted from image sequences.

Background Literally translates to "negative." A background image is a negative image, i.e., contains no positive samples to be extracted.

Window An imaginary rectangular outline that is moved around an image, containing the next set of pixels that will be extracted as a new sample frame.

Integral image A frame that has been converted to a special form described further in section 2.1.2

Positive image An image that contains the target object.

Negative image An image that does not contain any instance of the target object.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 2:
Background

Scientists have been striving to effectively mimic the human visual system on computers since the 1960s. The problem was severely underestimated at that time by Marvin Minsky, a professor at MIT, who asked his undergraduate student to "spend the summer linking a camera to a computer and getting the computer to describe what it saw" [5]. It was not until 30 years later that a computer could recognize a human face. Today, computer vision applications include automatically finding faces of your friends on Facebook, Google's robotic cars that autonomously navigate the hectic streets of San Francisco, and NASA's Curiosity rover that will explore Mars over the next decade. In each of these examples, object detection plays a vital role. Object detection is the process of distinguishing the target object from the rest of the image, e.g., your friend's face from the sunset in the background. A large advancement was made in this field by Paul Viola and Michael Jones in 2001. In their article [1], they described a visual object detection framework that was faster, more accurate, and more robust than anything at the time. This framework was duly well-received, and since then, many object detection algorithms, as well as many action detection and recognition algorithms, have been built off of the Viola-Jones system.

2.1 Viola-Jones Visual Object Detection Framework

The Viola-Jones visual object detection framework can be broken into four main components: Haar features, integral images, boosting, and cascades. Haar features are rectangular filters that extract information used for classification from sample images. The computation of Haar features is made significantly more efficient with integral images. All images are converted to this special format before any training or detecting takes place. Features are the building blocks for classifiers, which get arranged into dynamic groupings via the process of boosting. The final arrangement of classifiers is in the sequential form of a cascade, which quickly eliminates data easily classified as negative, saving computational resources for the the difficult samples.

2.1.1 Haar Features

Haar features are composed of adjacent rectangular areas that are applied to a window within a given image [6]. A single Haar feature will have at least two separate areas, from which all pixel intensities within are summed. The feature values are computed by taking the difference

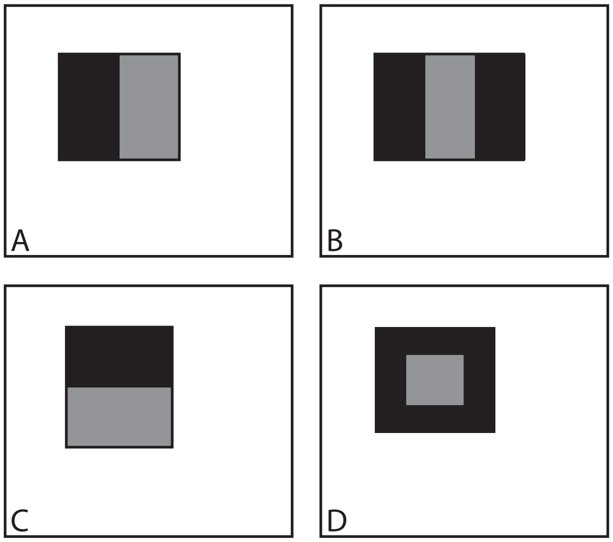

Figure 2.1: An illustrated example of Haar features.

between these summed areas. Figure 2.1 shows the geometries of the Haar features used in the OpenCV module from which our detector is built. "Tilted" Haar features [7] are also used in the OpenCV detector. They are created by simply rotating the geometries in Figure 2.1 by 45 degrees.

2.1.2 Integral Images

First introduced as "Summed-area tables" [8] by F. C. Crow in 1984, their usage did not become widespread until a reintroduction as "Integral images" by Viola et al. [1] in 2001. The integral image format significantly reduces the number of operations required for each feature computation, by adding to each pixel the sum of all of the pixels above and to the left of its position. Any x-y location in the integral image, contains the summed value of all the pixels within a rectangle that extends from the top-left corner of the image to that particular x-y location, as illustrated in Figure 2.2. This makes calculating any rectangular area of the image possible in only four matrix references as illustrated and described in Figure 2.3. This is important because the features used in the detection framework deal with only rectangular geometries as Figure 2.1 shows.

2.1.3 Classifiers

A classifier is the combination of a feature and its corresponding learned threshold value. A feature applied to an image sample results in a value that gets compared to a threshold. Anything

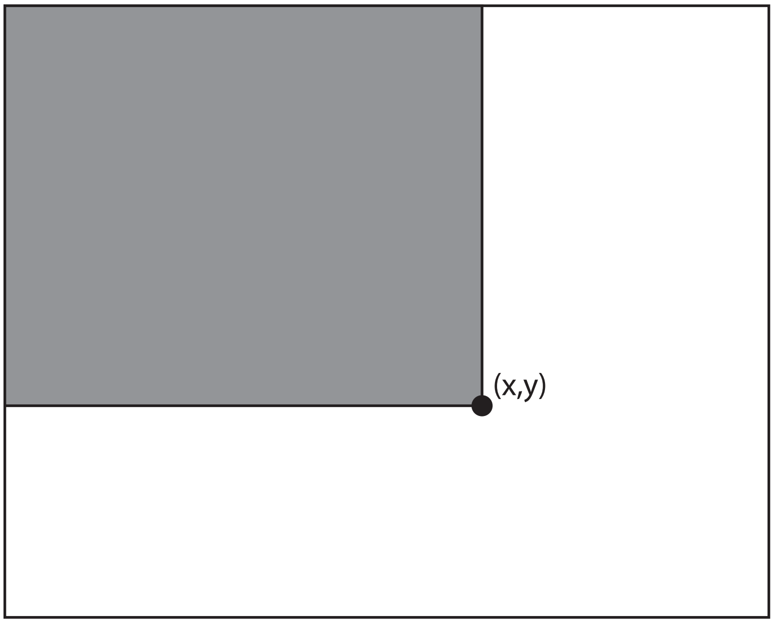

Figure 2.2: An illustrated example of integral images.

below that threshold is negative (i.e., no detection), and anything above it is positive, e.g., the classifier decides the object is present in the sample image.

The following equations [9] describe the classification process, beginning with applying a feature to an image sample which returns a feature value, v,

$$v_i = f_i(s) \tag{2.1}$$

where f_i is the ith feature and s is a single-frame sample. The feature value is then compared to a threshold, t:

$$c_i = \begin{cases} +1, & \text{if } v_i \geq t_i \\ -1, & \text{if } v_i < t_i \end{cases} \tag{2.2}$$

The classification c is set to one of two values, -1 or +1, representing a positive or negative evaluation, i.e., "object detected" or "no object detected." The classification values from multiple "weak" classifiers are then each multiplied by their own learned weight w and combined to

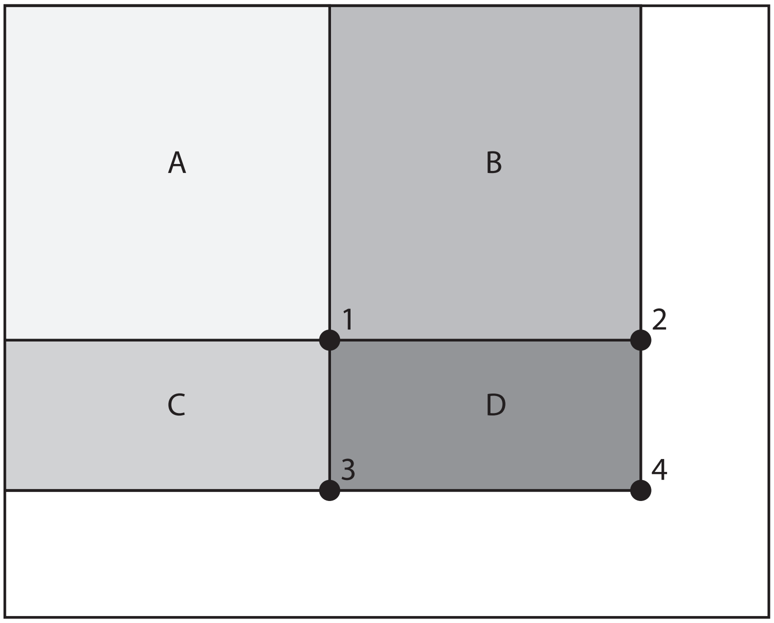

Figure 2.3: An illustrated example of an integral image. Area D can be computed in only four matrix references, $4-3-2+1$, which effectively subtracts areas A, B, and C from the area that extends from the top-left corner to point 4, leaving area D. Points 2 and 3 contain the values of $B+A$ and $C+A$, respectively. Thus, area A is effectively subtracted twice, hence, needs to be added back, once.

form a "strong" classifier, H,

$$H = \begin{cases} +1, & \text{if } \sum_{i=1}^{n} w_i c_i \geq 0 \\ -1, & \text{if } \sum_{i=1}^{n} w_i c_i < 0 \end{cases} \tag{2.3}$$

where n is the number of rounds of "boosting."

2.1.4 Boosting

Boosting is a machine learning process that builds a strong classifier by iteratively adding weak classifiers until a desired false-alarm rate is achieved. During this process, the weak classifiers are weighted according to their error rate. The Viola-Jones object detection framework [1] uses a form of AdaBoost [10], [11] to produce effective classifiers with relatively low numbers of features. A set of weak classifiers will be tested on training data to determine how good each one is at classifying the data correctly. The weak classifiers that best classify the data will be boosted to have more weight than the weak classifiers that do not perform as well. The classification results of a few select weak classifiers are linearly combined, with their respective boosted weights dictating the relative influence each has on the final classification. A helpful analogy is employing a group of expert analysts to make a yes-or-no decision. Some of those analyst's opinions will matter more than others and thus, have more influence on the final decision [12].

2.1.5 Cascades

Strong classifiers are then arranged in a cascade [1]. Commonly, they are arranged according to their number of weak classifiers, ascending in order. A sample must pass inspection by every strong classifier in the cascade before being classified as positive. A negative classification at any point in the cascade results in a final negative classification, as illustrated in Figure 2.4. This would be like having several groups of experts. The smallest groups of experts would be the first to classify the sample. If they reject it as negative, it stops there. The larger groups need not bother. This saves ample computation time. A significant amount of samples can be labeled as negative before even reaching the second strong classifier of the cascade. In the groups of analysts analogy, you would get the same results if you just applied one large group of all the experts to all the samples, but you would have to pay all of them for every sample to be classified. Using a cascade approach, you would only have to pay some of the analysts, most of the time. The cascade process saves computational resources by quickly eliminating data that is easily classified as negative, which happens to be the majority of the data when the target object is sparse, as it is in many cases.

2.2 Software Used

2.2.1 OpenCV

OpenCV is a highly optimized open-source computer vision library written natively in C++, which is the interface used in this thesis. It was originally developed by Intel, but is now supported by Willow Garage and Itseez [13]. The library includes many applications including facial recognition, mobile robotics, motion tracking, and artificial neural networks, that are employed by well-established companies like Google, Intel, and Honda for tasks such as stitching Google Streetview images, monitoring mine equipment in China, helping robots navigate and pick up objects, detection of swimming pool drowning accidents in Europe, and running interactive art in Spain and New York [13].

2.2.2 OpenCV Haartraining Module

The Haartraining module in OpenCV is an implementation of the Viola-Jones framework. The training algorithm takes as input a training set of positive and negative images, and produces

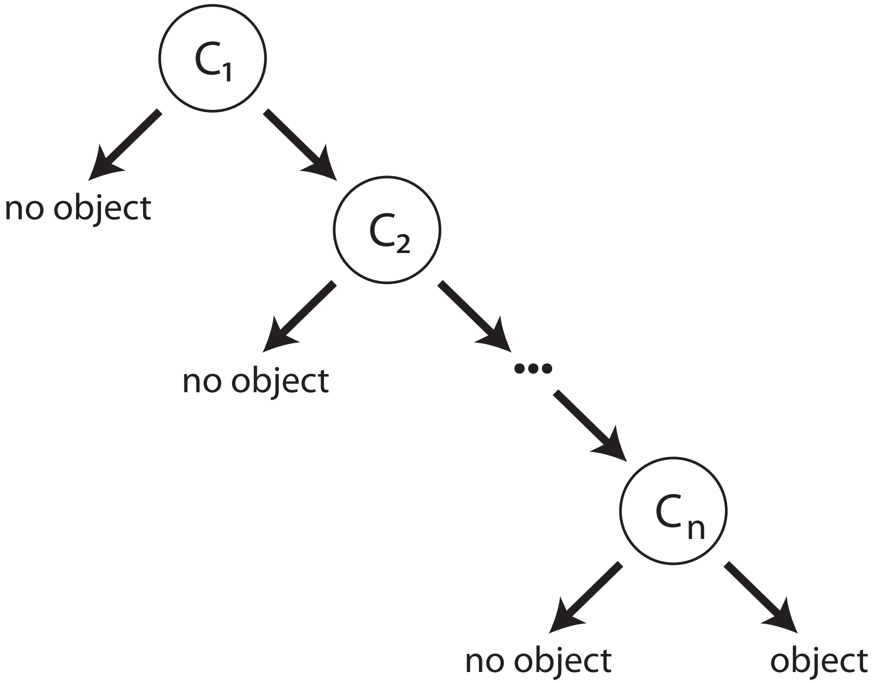

Figure 2.4: An illustrated example of a cascade of strong classifiers.

a cascade in the form of an xml file that can subsequently be used to detect the target object in images. As explained in Section 2.1.5, a cascade is a binary classifier that simply decides whether or not an object in an image is similar to the training set [9]. Other vital inputs to the training algorithm include:

Number of stages The maximum number of stages (a.k.a. strong classifiers) in the final cascade.

Hit rate The minimum rate at which each stage correctly classifies positive samples.

False-alarm (FA) rate The maximum rate at which each stage incorrectly classifies negative samples.

There is an interdependence between the number of stages, hit rate, and false-alarm rate, where the final cascade's hit and false-alarm rates are required to be as good as h^n and f^n, where h is the hit rate, f is the FA rate, and n is the number of stages. For instance, setting $h = 0.99$ and $n = 10$, ensures that the trainer will produce a cascade that achieves a 0.9044 hit rate $(= 0.99^{10})$ or better, or it will fail trying. The Viola-Jones framework builds strong classifiers, or "stages" in OpenCV, by iteratively adding weak classifiers. The haartraining algorithm does this until the hit rate *and* FA rate are met or exceeded for each stage.

CHAPTER 3:
Method

We initially considered two approaches for our method. The approach we decided against involved combining individual integral images into one combined integral image, as done by Ke et al. [14]. This way, volumetric features can be calculated on-the-fly very quickly, with a single cubic sum requiring only eight array references. This method is difficult to implement because the process of storing integral images and evaluating them with features must change, but it offers a big pay-off in terms of efficiency when the number of frames in the multi-frame samples grows large, e.g., the Ke et al. detector [14] uses samples that span up to 40 frames. While our detector does not limit the number of frames in a sample, we are mostly concerned with a smaller domain of two to five frames. The reason is that the characteristic motion of a pedestrian, car, animal, etc., would only span a few video frames on a typical 30fps camera and still contain enough information for accurate and robust detection. Viola-Jones already showed that there is enough information in just two frames for pedestrian detection [2]. Our chosen number-of-frames domain is not large enough to see significant gains in efficiency from combined integral images.

The approach we decided on was to separately apply two-dimensional features to each frame of a multi-frame sample in the same locations and sum the results, as shown in the following equation and illustrated in Figure 3.1,

$$v_i = f_i(F_1) + f_i(F_2) + f_i(F_3) + ... + f_i(F_n) \tag{3.1}$$

where f_i is the ith feature, F_1 is the first frame in the multi-frame sample, F_2 is the second, and so on. n is the total number of frames in the multi-frame sample. This fundamental change in the way the detector calculates feature values, as in Equation 2.1, is what allows the algorithm to access more information for every classification made. The summed value v is then compared to a threshold t, just as in Equation 2.2.

Features were computed over raw pixel intensities unlike Ke et al. [14] who hold that optical flow values are superior for action recognition and Viola et al. [2] who use various shifted difference images. Our application case is for a non-static camera, hence optical flow and

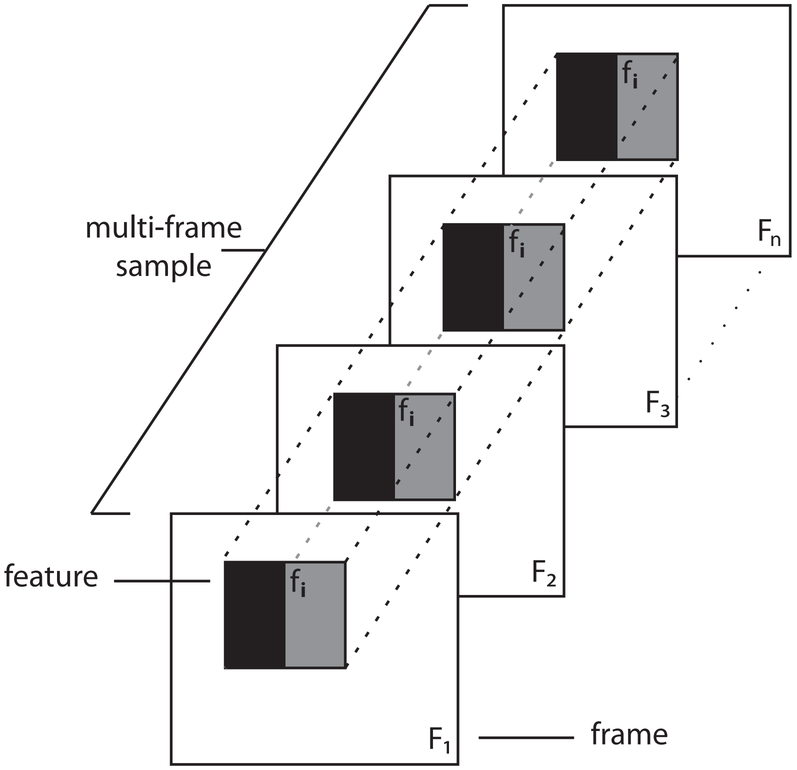

Figure 3.1: An illustrated example of a feature applied to a multi-frame sample

difference images become less attractive preprocessing steps. Gorelick et al. [4] achieve fast and robust action recognition with extracted and concatenated silhouettes, but we are less concerned with specific actions than with detection abilities at extremely low image resolutions where information is scarce. Discarding intensity information would seem to hurt our cause.

Instead of implementing our detector from scratch, we modified an existing implementation of the Viola-Jones visual object detection framework. The algorithm we used is part of the haartraining module from the OpenCV Library. Our multi-frame features could theoretically be applied to any implementation of the Viola-Jones visual object detection framework.

3.1 Implementing Multi-frame Feature Calculations

The main OpenCV haartraining algorithm is spread amongst several `C++` source and header files that make up the haartraining module. These files rely on core functions elsewhere in the OpenCV library. Feature calculations essentially take place in a single function within the haartraining algorithm. This function, namely `cvEvalFastHaarFeature`, is called multiple times throughout the algorithm, but every time a feature value gets computed it happens in this function. Our multi-frame feature-calculations were implemented within this function.

In the parts of the haartraining code that feature value calculations are called upon, the algorithm can be described as such:

```
For each single-frame sample:
        For each feature:
                featureValue = featureWeight * feature(frame);
```

where `feature(frame)` represents the resulting value of applying the current feature to the current single-frame sample.

Typically, a subset of single-frame samples are evaluated by a subset of features. During this process, a single-frame sample is evaluated by each and every feature, before moving onto the next single-frame sample.

Our modified version of the algorithm can be described as:

```
For each multi-frame sample:
        For each feature:
                For each frame in the current multi-frame sample:
                        featureValue = featureWeight * feature(frame);
                        summedValue += featureValue;
```

remember that in our modified detector a sample is no longer a single frame, but a sequence of multiple frames. A given feature is applied to every frame of a given sample and the results are summed, before the next feature is applied. This entire modification is accomplished in the scope of the `cvEvalFastHaarFeature` function, minimizing any necessary modifications to other parts of the haartraining algorithm. However, our modifications required reconsideration of feature-value normalization and extracting negative samples from background images. We feel subsequent modifications are worth describing since they would be necessary in almost any Viola-Jones framework implementation.

3.1.1 Feature Value Normalization

Before a weak classifier classifies an extracted feature value, the feature and threshold values are normalized according to the mean and variance of the intensity values of the pixels in the feature extraction area. This neutralizes the effects of the varying brightness and contrast among all extraction areas. Our modification to implement multi-frame feature-calculations led to modifying the process of normalization.

A normalization factor is calculated for each feature extraction area with the following equation:

$$normFactor_s = \sqrt{\frac{sqSum}{area} - \left(\frac{sum}{area}\right)^2}$$

where *area* is the number of pixels in the area under the feature, *sum* is the the sum of all pixel intensity values in the area under the feature, and *sqSum* is the sum of all squared pixel-intensity values in the area under the feature.

We could not simply leave the normalization factors to be calculated in this fashion, because it would create as many different normalization factors for a multi-frame feature extraction area as there are frames in the sample. The haartraining algorithm requires a single normalization factor. The weak classification process in the multi-frame detector is identical to that of the single-frame detector, shown in Equation 2.2. The multi-frame feature value is compared to a threshold, but first both need to be normalized by the normalization factor of that particular multi-frame feature extraction area. Separate normalization factors from each frame in the sample would be useless. A normalization factor needs to be calculated based on the average pixel intensity throughout the entire multi-frame feature extraction area. Here is the modified normalization equation:

$$normFactor_m = \sqrt{\frac{\sum_{i=1}^{n} sqSum_i}{\sum_{i=1}^{n} area_i} - \left(\frac{\sum_{i=1}^{n} sum_i}{\sum_{i=1}^{n} area_i}\right)^2}$$

where *n* is the number of frames in the multi-frame sample.

3.1.2 Extracting Negative Samples

Positive samples are cropped by the user prior to running the program, but the negative samples are automatically cropped by the algorithm from negative images. The "sliding window" approach used by the algorithm to extract negative samples from negative images needed to be modified for our multi-frame feature-calculations.

As positive and negative samples are read in, they are evaluated by the current cascade. This happens so the next stage can be trained on the tougher samples that the previous stages had misclassified. This does not pose a problem for the positive samples because they are already cropped and in the right order to be read in sequentially. However, the negative samples are not pre-cropped.

Originally, the haartraining algorithm extracts negative single-frame samples via the sliding-window approach. Here is a basic pseudocode of the process:

```
Do until the required number of negative single-frame samples is attained:
        Extract single-frame sample at current window location;
        If there is room to slide the window to the right:
                Slide window to the right;
        Else:
                Move window back to left side;
                If there is room to slide the window down:
                        Slide window down;
                Else:
                        Move window back to top;
                        Update scale factor;
                        If image can be rescaled:
                                Rescale image;
                        Else:
                                Load next negative image;
```

The problem is that a multi-frame sample needs to be extracted from a background *sequence*, thus the window should not be moved to the right or downward until it has first extracted a frame from each image of the current sequence. We solved this by reading in all images of the current background sequence and then proceeding to move the window inward, through the time dimension, before attempting to slide it to the right. Here is the previous pseudocode with our modifications added:

```
Do until the required number of negative multi-frame samples is attained:
        Extract multi-frame sample at current window location;
        If there is another frame:
                Move window to next frame;
        Else:
                Move window back to initial frame;
                If there is room to slide the window to the right:
                        Slide window to the right;
                Else:
                        Move window back to left side;
                        If there is room to slide the window down:
                                Slide window down;
                        Else:
                                Move window back to top;
                                Update scale factor;
                                If window can be rescaled:
                                        Rescale window;
                                Else:
                                        Load next negative image;
```

CHAPTER 4:
Experiment

In this chapter, we specify the details of the experiments conducted to obtain the results in Chapter 5. A synthetic dataset was created in Photoshop and used to train single- and multi-frame detectors that were then tested on the training set and compared in Chapters 5 and 6.

4.1 Dataset

We created a synthetic dataset that depicts a circular object undergoing a diagonal back-and-forth motion. 200 sequences, each spanning three frames, were created with the object superimposed over static background images. The background images were varied between sequences, as well as the appearance of the object in order to simulate random sensor noise and object variability. This was achieved in Photoshop by applying a layer mask filter of a random image on top of the object with varied opacity. The size and shape of the object were varied only slightly. Frames from some of the most varied samples are shown in Figure 4.1, on the right side of the dividing line.

From these sequences, positive training samples were extracted for both the single- and multi-frame detectors. All extracted frames are originally 30×30 pixels. The single-frame samples are extracted so that the object is centered both horizontally and vertically. The multi-frame samples are extracted so the object's motion across the three frames is contained without having to move the extraction area. In the multi-frame samples, the object moves from top-left, to bottom-right, to middle-center with significant overlap, and with a five pixel buffer from the edges. In order to compare the detectors at small scales, several sets were generated at smaller scales by uniformly decreasing the height and width dimensions of every frame for both the single-frame and multi-frame sample datasets. Examples are displayed in Figure 4.1. In all original samples, the object spans a rectangular area of at most 15×15 pixels. In the height- and width-reduced 3×3 samples, the object resolution varies from 2×2 to 3×3 pixels.

4.2 Training the Detectors

The training algorithm is parameterized with a minimum per-stage hit rate rate and a maximum per-stage false-alarm (FA) rate. During the training of each stage the algorithm iteratively adds weak classifiers until the FA rate is lowered below the maximum, while keeping above the

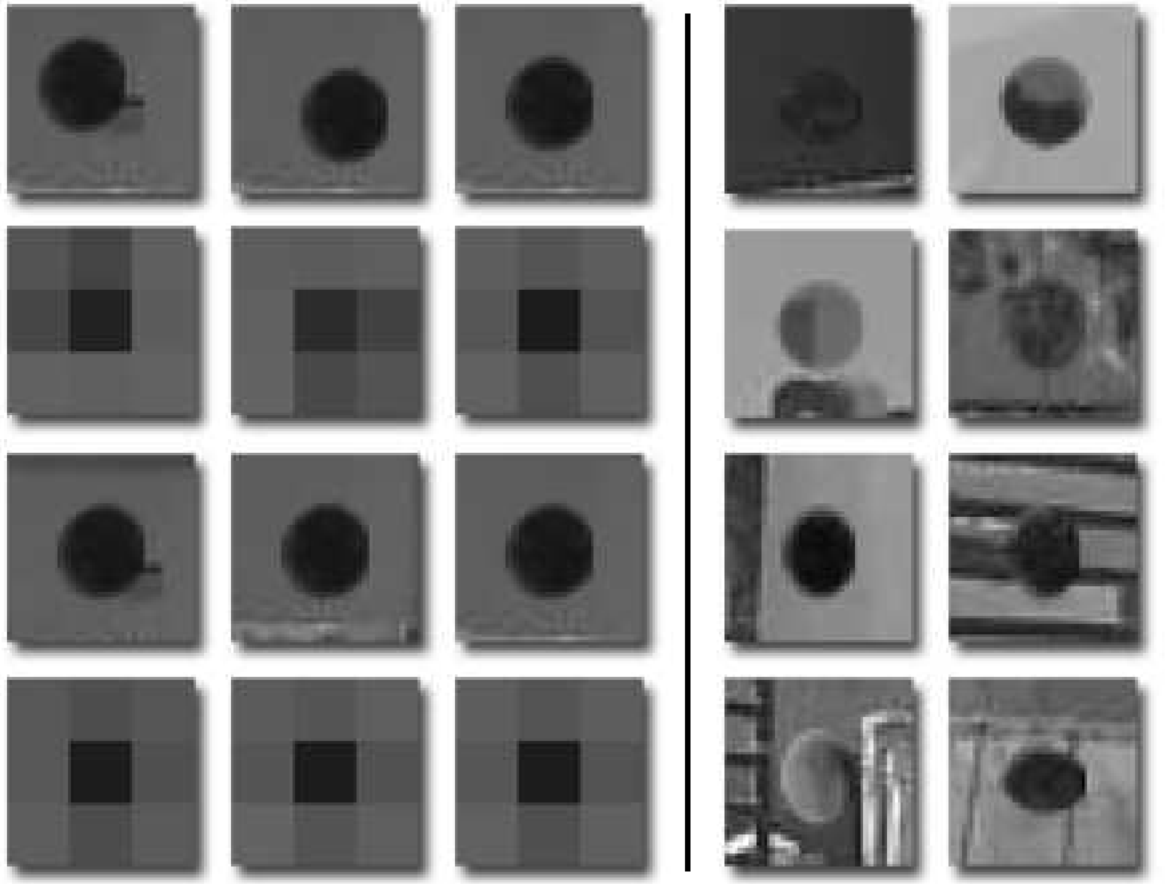

Figure 4.1: Examples of single- and multi-frame samples. The top row, left of the line, contains the original-size, 30×30, multi-frame sample, with the scaled-down, 3×3, version in the next row, down. The last two rows show the corresponding single-frame sample at 30×30 and 3×3 pixels. All frames right of the line are displayed to demonstrated the variability between samples.

minimum hit rate. After all stages are trained, the algorithm performs an evaluation on the training set with the finalized detector that results in overall hit and FA rates, which are the rates we used to compare the detectors in Figure 5.1.

The single-frame detectors were trained on 450 single-frame positive samples and 900 single-frame negative samples, while the corresponding multi-frame detectors were trained on 150 three-frame positive samples and 900 three-frame negative samples.

4.2.1 Ensuring Comparability

Certain measures had to be taken with the datasets to ensure the training results between the single- multi-frame detectors were accurately comparable. While the number of samples differed, the total number of frames in a multi-frame dataset always contained the same number of total frames as the corresponding single-frame set. For example, if the multi-frame detector was trained on a set of 100 three-frame samples, the single-frame detector was trained on 300 single-frame samples that corresponded to the multi-frame samples, as in Figure 4.1. This way, both detectors are trained on the same amount of total information. The number of background samples used for training was equal between single- and multi-frame detectors. Each multi-frame background sample was created by combining three copies of a single-frame sample.

4.2.2 Stage Training

The number of weak classifiers per stage is a useful measurement because it shows when stage training has struggled to meet the maximum FA rate requirement. As described in the previous paragraph and Sections 2.1.4 and 2.1.3, weak classifiers are iteratively added until the maximum false-alarm rate is achieved, so the more difficult the data is to correctly classify for a given detector, the more weak classifiers will be added. If a cascade stage simply cannot reach the maximum FA rate, the process of adding weak classifiers will cycle indefinitely until it is killed by the user. Thus, a final cascade cannot be produced, and the trainer has failed at the specific hit and FA rate requirements.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 5:
Results

In this chapter, we present the results of the experiments described in Chapter 4. Results from detectors trained on data at the 3×3 pixel scale are plotted as a receiver operating characteristic (ROC) curve in order to display the differences in detection rates. Results from detectors trained on the 4×4 pixel scale are plotted as FA rate versus weak classifiers per stage in order to show any difficulties in the training processes.

The multi-frame detector exceeded the capabilities of the single-frame detector by classifying the training data with FA rates significantly less than the single-frame detector's limit of 0.62. The single-frame training algorithm failed at attempts to produce a detectors with FA rates below 0.62, despite decreasing the minimum stage hit rate below 0.50. The single-frame training algorithm was able to produce a detector with an FA rate of 0.626, but with a hit rate (0.820) approximately 17% lower than what the multi-frame detector achieved (0.993) with an approximately equal FA rate. The complete hit and FA rate results for the 3×3 training set are displayed in Figure 5.1.

5.1 ROC Curve

We produced an ROC curve from the output results of the training process by steadily varying the maximum per-stage FA rate, from 0.75 to 0.95, and raising the hit rate until the detectors failed to decrease the FA rate below the maximum. This produced the best hit rate achievable for a given maximum FA rate. It is important to understand that, here the "given maximum FA rate" is for stage training, and the overall FA rate achieved by the final cascade is the value used to produce the ROC curve. The ROC curve is displayed in Figure 5.1.

The multi-frame detector outperformed the single-frame detector at the 3×3 scale by not just achieving better detection rates, but by producing any detectors at all at lower maximum per-stage FA rates. Comparing the one data point produced by the single-frame detector to the multi-frame detector's ROC curve, the multi-frame detector has a hit rate that is approximately 20% higher than the single-frame detector (a rate difference of approximately 0.15). Perhaps more importantly, multi-frame detectors can be trained at the 3×3 scale with FA rates as low as 0.302, which was not possible for single-frame detectors due to training failure.

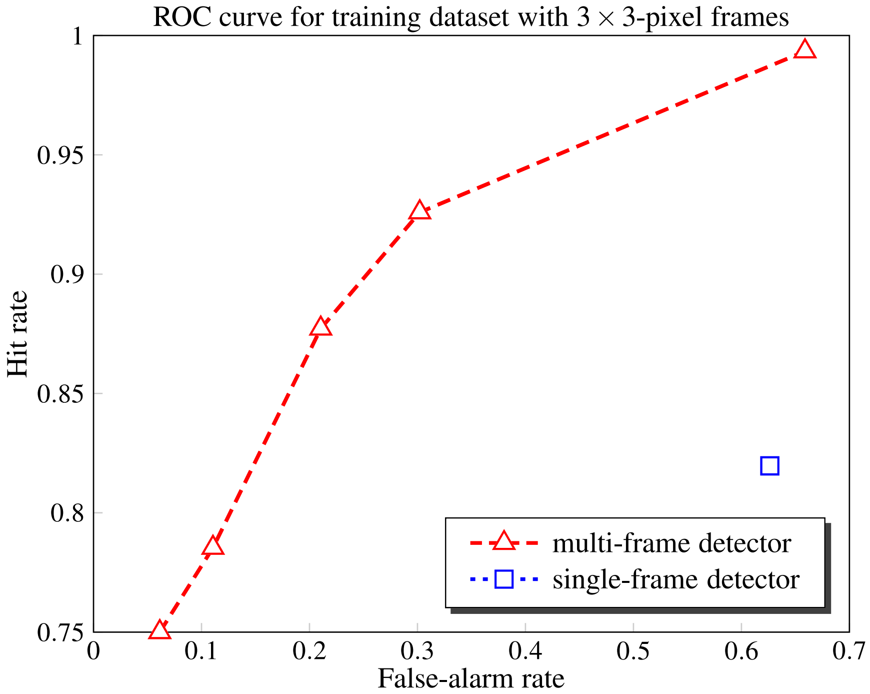

Figure 5.1: ROC curve for single- and multi-frame detector performance on a training set with a sample frame-size of 3×3 pixels.

5.2 Weak Classifiers per Stage

Next, we trained detectors on a slightly larger (4×4 pixel) dataset with the same number of positive and negative samples. Both detectors were able to be trained to classify the data very accurately, with perfect hit rates of 1.0 and FA rates below 6.6×10^{-4}. With both detectors achieving such a low FA rate, we focused on the number weak classifiers being produced by the detectors in order to further examine and compare at the 4×4 scale.

The maximum per-stage FA rate parameter was varied from 0.3 to 0.8, while a perfect hit rate of 1.0 was held constant. To simplify the detector a bit, the number of stages for each detector was lowered from ten, which was used for the 3×3 scale, to seven. The effect of this change was a decreased performance in the overall completed cascade detection rates, but not in the per-stage rate limits. The number of weak classifiers that were required for each stage were summed and divided by the number of stage classifiers to get the the number of weak classifiers per stage in the cascade. Figure 5.2 displays the resulting plot.

In Figure 5.2, we see that the single-frame detector, on average, requires significantly more

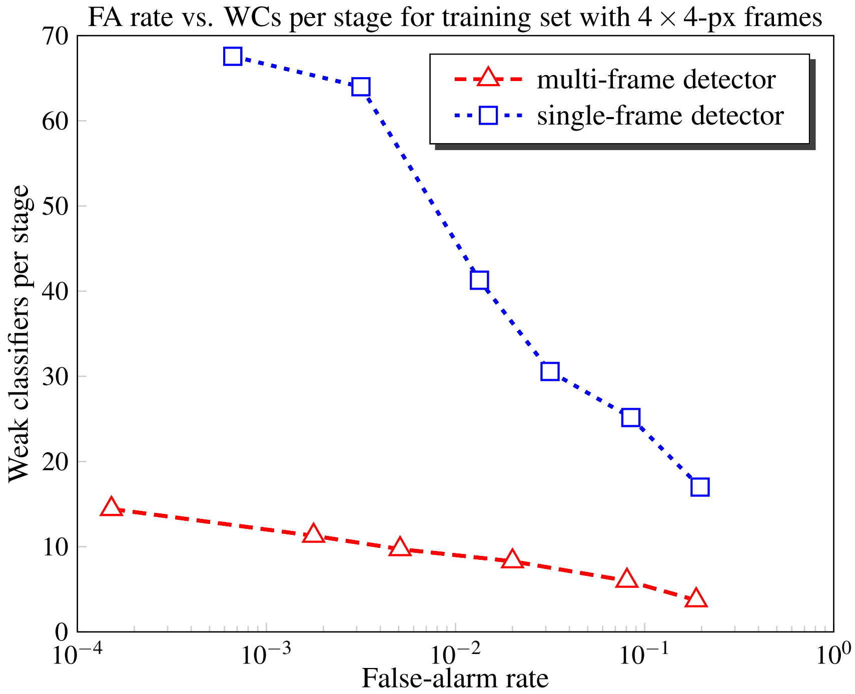

Figure 5.2: FA rate vs weak classifiers (WCs) per stage required in the cascade with a sample frame-size of 3×3 pixels. A perfect final hit rate of 1.0 was held constant.

weak classifiers per stage than for the multi-frame detector at the same FA rates. Furthermore, the difference between the two grows at a substantial rate as the FA rate approaches zero.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 6:
Discussion

Our hypothesis that extending the Viola-Jones detector to extract information from multiple frames will improve detection capabilities at extremely small scales was supported by our results. We have shown that the multi-frame detector succeeds on scales too small for the single-frame detector. However, our experiments have significant limitations such as not utilizing a separate test set and training only on synthetic, three-frame image sequences.

6.1 Training Results

When the training process fails, it means the feature values were not separable enough, for the given detection rate requirements, into positive and negative groupings because there was not enough information that could consistently distinguish the target object from the background. This can be a result of too much variability between instances of the target object in positive samples, i.e., the variability between samples of the target object was just as dynamic as the variability between the target object and the background. It could be the case that there's minimal variability between positive samples, but simply not enough information on the target object to sort it from the background. For instance, in most cases a detector could never learn to separate a target object of one pixel from the background, because there will exist many single pixels in the background with similar intensity, despite them not corresponding to a true target object. This was the case for the single-frame detector training on the 3×3 pixel samples. There was not enough information within those nine pixels to distinguish the target object from the background with a false-alarm rate less than 0.62. The additional motion information used by the multiple-frame detector was enough to distinguish the target from the background with incorrectly classifying negative samples as positive at rates below 0.63, where the single-frame detector failed.

The FA versus weak classifiers per stage plot supports these results by showing that the single-frame detector was struggling to accurately classify the 4×4 pixel version of the training dataset. It required a very large number of weak classifiers per stage to meet the same detection rates achieved by the multi-frame detector with significantly less weak classifiers per stage.

Our findings show the multi-frame detector exceeds the performance of the single-frame detector on a training dataset that contains very low resolution (3×3 pixels) on the target object. This suggests that the multi-frame detector could excel on other datasets that may be too challenging for the single-frame detector by lacking in either the target object resolution or the commonality in the appearance of the object among the positive sample set. The requirement for our detector to exceed the single-frame detector is that there be useful information related to the motion of the object. Most importantly, the motion of the object must have commonality within the range of a few frames. In our case, the object repeated only one motion in each sample. We expect that, just as with the object appearance, the more varied the motion is, the less useful that motion information will be to the detector, and the less advantage our detector will have over over a single-frame detector.

6.2 Limitations and Future Work

The detectors were not tested on a dataset separate from the training set. The detectors were trained and then tested on the same datasets on which they were trained. In real world applications, test sets will be separate from the training set, thus introducing a new element of variability that will depend on the type of target object. It is important for future work and applications that this type of testing occur, to obtain a reliable measure of accuracy and robustness. Another limitation is that our experiments only included data from one type of imagery, which was synthetically constructed and contained sequences spanning only three frames. Future work and testing should include natural image datasets, differing numbers of frames per sample, and other target objects with more than one type of characteristic motion. Finally, the FA rate achieved by the multi-frame detector of 0.3 at the 3×3 pixel scale is high for most practical applications. For this reason, as well as the aforementioned limitations, it has yet to be proven that our detector provides a significant benefit in terms of practical applications.

CHAPTER 7:
Conclusion

We have extended the Viola-Jones detection framework to operate at extremely small scales, where the original version struggles to classify the same data, and in some cases fails completely. By utilizing information related to motion in addition to appearance, our multi-frame detector accurately detects objects over three consecutive video frames at scales smaller than is possible with appearance information contained within a single frame.

Our modification to the Viola-Jones visual object detection framework is a fundamental change in the way the feature values are calculated, that implements a method to combine information from multiple sequential video frames. Integrating this information from three frames, as opposed to one [1] or two frames [2], allows our detector to extract information related to a motion that is characteristic of the target object. In our experiment, our detector utilizes raw pixel-intensities. Pre-calculated forms such as optical flow or difference images are compatible with our framework, but not employed in this thesis.

We suggest that our multi-frame detector might also be applicable to video datasets of objects at normal scales that are typically hard to detect because of large variation in appearance, but exhibit characteristic motions within a few consecutive video frames. Areas where this type of small scale detector might be useful include full-motion video processing where the target object spans an area of 4×4 pixels or less. Possible application areas may include full-motion video from satellite, airborne, or ground-based sensors. The range of target objects to which our detector can be applied is yet to be determined. We suggest pedestrian detection as a viable application due to inherent characteristic motion. Our experiment was done with a common three-frame motion. We also suggest that objects undergoing characteristic motions that span four or five frames might be good candidates for our multi-frame detector, without incurring an overwhelming loss of computational efficiency.

THIS PAGE INTENTIONALLY LEFT BLANK

List of References

[1] P. Viola and M. Jones, "Robust real-time object detection," *International Journal of Computer Vision*, vol. 57, no. 2, pp. 137–154, 2001.

[2] P. Viola, M. Jones, and D. Snow, "Detecting pedestrians using patterns of motion and appearance," *International Journal of Computer Vision*, vol. 63, no. 2, pp. 153–161, July 2005.

[3] L. Lee, "Gait dynamics for recognition and classification," in *AI Memo 2001-019*. Cambridge, MA: MIT Press, 2001.

[4] L. Gorelick, M. Blank, E. Shechtman, M. Irani, and R. Basri, "Actions as space-time shapes," *Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 12, pp. 2247–2253, December 2007.

[5] R. Szeliski, *Computer Vision: Algorithms and Applications*. http://szeliski.org/Book/: Springer, 2009.

[6] C. P. Papageorgiou, M. Oren, and T. Poggio, "A general framework for object detection," in *Proceedings of the Sixth International Conference on Computer Vision*, ser. ICCV '98. Washington, DC, USA: IEEE Computer Society, 1998, pp. 555–562.

[7] R. Lienhart and J. Maydt, "An extended set of haar-like features for rapid object detection," in *IEEE ICIP 2002*, 2002, pp. 900–903.

[8] F. C. Crow, "Summed-area tables for texture mapping," *SIGGRAPH Comput. Graph.*, vol. 18, no. 3, pp. 207–212, jan 1984.

[9] G. Bradski and A. Kaehler, *Learning OpenCV: Computer Vision in C++ with the OpenCV Library*, 1st ed., M. Loukides, Ed. Sebastopol, CA: O'Reilly Media, 2008.

[10] Y. Freund and R. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," in *Computational Learning Theory: EuroCOLT '95*, vol. 55, no. SS971504. London, UK: Springer-Verlag, 1995, pp. 23–37.

[11] R. Schapire, Y. Freund, P. Bartlett, and W. S. Lee, "Boosting the margin: A new explanation for the effectiveness of voting methods," in *The Annals of Statististics*, vol. 26(5), 1998, pp. 1651–1686.

[12] R. Rojas, “Adaboost and the super bowl of classifiers: A tutorial introduction to adaptive boosting,” December 2009, unpublished tutorial.

[13] “About opencv,” http://opencv.org/about.html, accessed: 28/07/2012.

[14] Y. Ke, R. Sukthankar, and M. Hebert, “Efficient visual event detection using volumetric features,” in *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, vol. 1. Intel Research, October 2005, pp. 166–173.

Initial Distribution List

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudly Knox Library
   Naval Postgraduate School
   Monterey, California

3. Mathias Kolsch, PhD
   Associate Professor
   Department of Computer Science
   Naval Postgraduate School
   Monterey, California

4. Richard C. Olsen, PhD
   Professor
   Department of Physics
   Naval Postgraduate School
   Monterey, California

5. Dan C. Boger, PhD
   Naval Postgraduate School
   Monterey, California